In the claims:

The following proposed amendment of claims was to be decided by the examiner after reconsidering the Remark provided herein.

Proposed amendment of claims:
Please amend the subject claims as follow:

86. (Currently amended) A ~~programming tool~~ system configured for a programmer to program a first computing device comprises at least a first and a second table, wherein said first table comprises a first plurality of configuration states, at least one of said configuration states defining one or more qualifying conditions; wherein

said second table is configured to specify a second plurality of paths, at least one of said paths being executed when a qualifying condition listed in said first table is satisfied; and one of said configuration states being specified as an active state; characterized in that said ~~tool~~ system further comprises one or more of the following elements or characteristics:

(1) at least one of said configuration state comprising a virtual qualifier;

(2) at least one of said second plurality of paths having one or more labels, wherein each label represents an executable program;

(3) one or more further tables specifying user-defined custom expressions to represent corresponding predefined instructions of a programming language;

(4) said first computing device being a remote computer connecting to a second local computing device through a communication link; wherein the digital data representing said first and second tables is stored in said remote computing device for downloading to said local computing device through said communication link;

(5) said first computing device comprising at least a first processor and a second processor; wherein said first processor is configured to translate the relationship between said first and second tables and to give direction to the second processor to execute other programs;

(6) at least one of said configuration states comprising a label and said label being equated with a separate expression defining the qualifying condition represented by said label;

(7) a table defining a preferable style of keywords;

(8) a table having a third plurality of task states to define the activity of a fourth plurality of tasks;

(9) a table defining a fifth plurality of output expressions, each output expression representing an output condition of a configuration state or a path element listed in said second table.

(10) a further state table defining a sixth plurality of configuration states and a further path table defining a seventh plurality of paths; wherein said first and second tables are grouped as a first table group to perform a first function, said further state and path tables are grouped as a second table group to perform a different second function;

~~Wherein~~ said ~~programming tool~~ system is further configured to transform the configuration states and paths programmed by said programmer into codes executable by said first computing device; and

wherein said system comprises at least one of the following hardware:

(a) a computing device configured to compile said first and second tables;

(b) a tool or work station provided to compile said first and second tables;

(c) a computing running the task compiled from said first and second tables;

(d) a debugging hardware;

(e) a communication link;

(f) a printer configured to print a program represented by said first and second tables; and

(g) a display device configured to display elements of said first or second tables.

87. (Currently amended) The ~~programming tool~~ system of claim 86 wherein said tables and/or elements are configured by a programmer in accordance to a preprogram objective, and to be translated into a program executable by said computing device to deliver said preprogram objective.

88. (Currently amended) The ~~programming tool~~ system of claim 86 wherein ~~said programming tool comprises a third computing device having a table format compiler configured to compile a table format program input into said third computing device to generate codes executable by said first computing device~~ the computing device of hardware (a) compiles said first and second tables into codes executable by said first computing device.

89. (Previously presented) A system comprising a computer accessible memory means or medium storing a computer program; wherein said computing program is configured by a programmer in accordance to a predefined programming objective for executing by a computing device to deliver said programming objective; said computing program further comprises a first table having x configuration states, at least one of said configuration states is configured to define one or more qualifying conditions;

a second table specifying y paths, at least one of said paths is executed by said computing device when a qualifying condition enlisted in said first table is satisfied; said computing program further comprises at least one of the following tables:

(1) a table specifying user defined custom expressions to equate the corresponding predefined instructions of a programming language;

(2) a table defining preferable style of keywords;

(3) a table having m task states to define the activity of n tasks, at least one of said task states specifies k selected tasks to be active;

(4) a table having p task states to define q tasks, at least one of said task states specifies the priority of the active tasks to be serviced;

(5) a table defining x qualifying expressions, each qualifying expression represents a qualifying condition of a configuration state; and

(6) a table defining y output expressions, each output expression represent an output condition of a configuration state or a path element enlisted in said second table.

90. (Previously presented) The system of claim 89 further comprising a compiler provided for compiling said computer program into data executable by said computing device.

91. (Previously presented) The system of claim 90 wherein said data is further embedded in said computing device and said computing device is installed in an article of sale.

92. (Currently amended) A programming method for a programmer to program a computing device comprises the steps of:

(1) providing a ~~programming tool~~ system for said programmer to specify x configuration states;

(2) specifying one or more qualifying conditions to at least one of said x configuration states;

(3) configuring said ~~programming tool~~ system to specify y paths to be executed by said computing device;

(4) assigning z labels to represent one or more of the path elements of step ~~(3)~~ wherein z is an integer equal or greater than one;

(5) for each qualifying condition of step (2), specifying a path to be executed by said computing device when a specific qualifying condition is satisfied;

(6) specifying at least one of the configuration states to become the active configuration state;

(7) for each assigned label of step (4), defining an executable program to be represented by said label; ~~and~~

(8) transforming the configuration states and paths programmed by said programmer into codes executable by said first computing device; and

(9) providing at least one of the following hardware to work with said steps (1) to (8):

(a) a compiling computing device configured to compile said configuration states and paths;

(b) a tool or work station provided to compile said configuration states and paths;

(c) a computing running the task compiled from said configuration states and paths;

(d) a debugging hardware;

(e) a communication link;

(f) a printer configured to print a program represented by said configuration states and paths; and

(g) a display device configured to display elements of said configuration states and paths.

93. (Original) The programming method of claim 92 wherein the label of step (4) is not initially executable by said computing device and the additional step (7) is configured to enable the execution of said label by said computing device.

94. (Original) The programming method of claim 93 further comprising a step to identify a label not executable by said computing device for the composing of the program of step (7) thereof.

95. (Currently amended) The programming method of claim 92 wherein the program of step (7) is composed by any available programming languages ~~including the~~ or a table format programming language.

96. (Original) The programming method of claim 95 further comprising a step to define means to pass parameters between the program represented by the steps (1) to (6) and the executable program represented by the label.

97. (Original) The programming method of claim 92 further comprising a step to specify one or more output conditions to at least one of said x configuration states.

98. (Original) The programming method of claim 92 wherein at least an output condition is specified in a path or in a configuration state.

99. (Original) The programming method of claim 92 further comprising a step to specify resources of said computing device required to service the states and paths thereof.

100. (Original) The programming method of claim 92 further comprising a step to indicate the resources of said computing device available for composing the program of step (7).

101. (Original) The programming method of claim 92 wherein said computing device comprises of two or more processors; the resources to service the states and paths specified are provided by a first processor and at least part of the program of step (7) is serviced by the resources of a second processor.

102. (Currently amended) The programming method of claim 92 wherein said computing device is defined as a first computing device; said programming method further comprising a step to receive digital data representing the aforementioned steps

through a the communication link of step (9)(e) from a second computing device locating remote from said first computing device.

103. (Original) The programming method of claim 92 wherein at least part of said steps are organized into a table format.

104. (Original) The programming method of claim 103 further comprising a step to group the configurations states of step (1) and (2) into one or more tables.

105. (Original) The programming method of claim 103 wherein the configuration states or paths are not necessary to be enlisted in sequential relationship to each other.

106. (Original) The programming method of claim 92 further comprising a step to transform the specifications of the configuration states and the paths into digital data to be stored into said computing device for the execution thereof.

107. (Original) The programming method of claim 92 further comprising a step to transform at least part of the specification of said steps with a secondary programming language of different format.

108. (Original) The programming method of claim 92 further comprising a step to identifying the location of the program composed by said steps.

109. (Original) The programming method of claim 95 wherein the program of step (7) is a program locates external to the program composed by said steps.

110. (Original) The programming method of claim 109 further comprising a step to identify the location of said external program.

111. (Original) The programming method of claim 92 further comprising a step to organize a separated table to specify if the program composed by steps (1) to (7) is active or not active.

112. (Original) The programming method of claim 92 further comprising a step to assign a label to represent a configuration state.

113. (Original) The programming method of claim 92 further comprising a step to assign a label to represent a path.

114. (Currently amended) A programming method for a programmer to program a computing device executing multiple programs, said programming method comprises the steps of:

(1) providing a ~~programming tool~~ system for said programmer to specify n programs executable by said computing device;

(2) defining m task states, each task state specifies k selected programs to be active, wherein k is an integer equal or greater than zero;

(3) specifying one of the task states of step (2) to be the active task state; ~~and~~

(4) transforming said m task states and k selected programs into codes executable by said computing device; and

(5) providing at least one of the following hardware to work with said steps (1) to (4):

(a) a compiling computing device configured to compile said task states;

(b) a tool or work station provided to compile said task states;

(c) a computing running the task compiled from said task states;

(d) a debugging hardware;

(e) a communication link;

(f) a printer configured to print a program represented by said task states; and

(g) a display device configured to display elements of said task states.

115. (Original) The programming method of 114 wherein at least one of the programs is a table format program comprising:

x program configuration states, at least one of said configuration states defines one or more qualifying conditions;

y paths to be executed by said computing device; and

one of said paths is executed by said computing device when a specific qualifying condition is satisfied.

116. (Original) The programming method of claim 114 wherein the programs of step (1) are of different languages.

117. (Original) The programming method of claim 114 further comprising a step to provide a keyword to represent the steps (1) to (3).

118. (Original) The programming method of claim 114 wherein said task states are not necessary to be listed in sequential relationship to each other.

119. (Original) The programming method of claim 114 wherein the steps (1) to (3) define a first task table, said programming method further comprising steps to define a second different task table.

120. (Currently amended) A programming method for a programmer to program a remote computing device, said programming method comprises the steps of:

(1) providing a ~~programming tool~~ system for said programmer to specify x configuration states, at least one of said configuration states defines one or more qualifying conditions;

(2) specifying y paths to be executed by said computing device;

(3) for each qualified condition of step (1), further specify a path to be executed by said computing device when a specific qualifying condition is satisfied;

(4) specifying one of the qualifier configuration states to become the active configuration state;

(5) storing digital data representing the configuration states and paths of the aforementioned steps in a local computing device;

(6) downloading the digital data of step (5) to said remote computing device though a communication link; and

(7) transforming the configuration states and paths programmed by said programmer into codes executable by said remote computing device; and

(8) providing at least one of the following hardware to work with said steps (1) to (7):

(a) a compiling computing device configured to compile said configuration states and paths;

(b) a tool or work station provided to compile said configuration states and paths;

(c) a computing running the task compiled from said configuration states and paths;

(d) a debugging hardware;

(e) a communication link;

(f) a printer configured to print a program represented by said configuration states and paths; and

(g) a display device configured to display elements of said configuration states and paths.

121. (Currently amended) The method of claim 120 wherein said communication link of step (8)(e) is a network.

122. (Original) The method of claim 121 wherein said network comprises of the internet; intranet extranet or LAN.

123. (Original) The method of claim 120 further comprising a step to direct an element of a path to a program written in a second programming language of different format.

124. (Original) The method of claim 120 further comprising a step to evaluate the architecture of said remote computing device and a further step to configure the aforementioned steps to work with the architecture of said remote computing device.

125. (Original) The method of claim 120 further comprising a step to organize at least part of the data specified by said steps into a table format.

126. (Original) The method of claim 120 further comprising a step to store digital data representing said configuration states and paths into said remote computing device for the execution thereof.

127. (Original) The method of claim 120 further comprising a step to transform at least part of the specifications of the configuration states and the paths into a secondary programming language of different format.

128. (Original) The method of claim 120 wherein the configuration states or paths are not necessary to be listed in sequential relationship to each other.

129. (Original) A multiple processors computing device comprising a first processor and a second processor wherein said first processor is configured to execute at least part of a table format program defined by m configuration states interact with n paths.

130. (Original) The multiple processors computing device of claim 129 wherein the second processor executes programs written with a second language not in table format.

131. (Original) The multiple processors computing device of claim 130 wherein the second processor is configured to execute a program as directed by a table format program executed by said first processor.

132. (Original) The multiple processors computing device of claim 129 wherein said first and second processors are included in a single integrated circuit.

133. (Original) The multiple processors computing device of claim 129 wherein at least one instruction executable by one of the processors is different from the instruction set executable by the other processor.

134. (Original) The multiple processors computing device of claim 129 further comprising means to pass parameter between said first and second processors.

135. (Currently amended) A method to compose a compiler suitable for a ~~programming tool~~ system to compile a table format program having m configuration states and n paths, and to generate a program or codes suitable to be executed by a computing device, comprising the steps of:

(1) defining a table format programming specification;

(2) identifying the region representing the configuration states;

(3) identifying the region representing the paths;

(4) identifying at least one qualifying condition of a configuration state and link it to the path specified;

(5) linking a configuration state A to a path quoting said configuration state A as its element; ~~and~~

(6) providing means for said programming tool to transform said configuration states and paths into codes executable by said computing device; and

(7) providing at least one of the following hardware to work with said steps (1) to (6):

(a) a compiling computing device configured to compile said configuration states and paths;

(b) a tool or work station provided to compile said configuration states and paths;

(c) a computing running the task compiled from said configuration states and paths;

(d) a debugging hardware;

(e) a communication link;

(f) a printer configured to print a program represented by said configuration states and paths; and

(g) a display device configured to display elements of said configuration states and paths.

136. (Original) The method of claim 135 further comprising a procedure to integrate the function of steps (1) to (4) into a compiler of an existing language.

137. (Original) The method of claim 135 further comprising a procedure in step (1) to identify a keyword representing the starting of the configuration states.

138. (Original) The method of claim 135 further comprising a procedure in step (2) to identify a keyword representing the starting of the paths.

139. (Currently amended) The method of claim 135 further comprising a step to equate an user defined expression with a specific instruction of ~~the~~ a table format programming language.

140. (Currently amended) The method of claim 139 further comprising a step to identify a table equating user define expressions with predefined instruction set of said table format programming language.

141. (Currently amended) The method of claim 135 further comprising a step to identify label unexecutable according to the instruction set of said table format ~~program~~ programming language.

142. (Original) The method of claim 141 further comprising a step to link said unexecutable label to an external program.

143. (Currently amended) The method of claim 142 wherein said external program comprises of a program composed by any available programming language including ~~the~~ a table format programming language.

144. (Original) The method of claim 135 further comprising a step to distinguish two or more program groups, wherein each program group comprises of at least one configuration state table and one path table.

145. (Original) The method of claim 144 further comprising a step to compile the interaction in between said multiple program groups.

146. (Original) The method of claim 142 further comprising a step to define an instruction to activate a program group.

147. (Original) The method of claim 146 wherein said instruction is represented by one or more task tables.

148. (Currently amended) A method for a programmer to program a computing apparatus comprising the steps of:

(1) providing said programmer a ~~programming tool~~ system configured to work with a programming language having a predefined instruction set;

(2) defining alternate expression to represent a specific instruction of the selected programming language;

(3) composing program with said alternate expression; and

(4) transforming the program of step (3) into codes executable by said computing apparatus; and

(5) providing at least one of the following hardware to work with said steps (1) to (4):

(a) a compiling computing device configured to compile the program of step (3);

(b) a tool or work station provided to compile the program of step (3);

(c) a computing running the task compiled from the program of step (3);

(d) a debugging hardware;

(e) a communication link;

(f) a printer configured to print a program represented by the program of step (3); and

(g) a display device configured to display elements of the program of step(3).

149. (Previously presented) The method of claim 148 wherein said programming language is a table format programming language comprising:

x configuration states, at least one of said configuration states defines one or more qualifying conditions;

y paths to be executed by said computing device; and

one of the paths is executed by said computing device when a specific qualifying condition is satisfied.

150. (Original) The method of claim 148 wherein the process of step (4) is a translation process carried out by an editor, a compiler, an interpreter or a translation program.

151. (Previously presented) The method of claim 150 further comprising the step of displaying the composed program with the translated predefined instruction set of said programming language.

152. (Original) The method of claim 150 further comprising the step of displaying the composed program with the user defined expressions.

153. (Original) The method of claim 148 further comprising a step to provide a table located inside the program for linking said alternate expression with the corresponding specific instruction.

154. (Currently amended) A programming method for a programmer to program a computing device comprises the steps of:

(1) providing a programming tool for said programmer to define one or more virtual qualifiers;

(2) specifying x configuration states, at least one of said configuration states defines one or more qualifying conditions;

(3) defining at least one of the qualifying conditions of step (2) to represent a virtual qualifiers;

(4) specifying y paths to be executed by said computing device;

(5) for each qualifying condition of step (2), further specify a path to be executed by said computing device when a specific qualifying condition of said qualifier is satisfied;

(6) specifying one of the qualifier configuration states to become the active configuration state; ~~and~~

(7) transforming said configuration states and paths into codes executable by said computing device; and

(8) providing at least one of the following hardware to work with said steps (1) to (7):

(a) a compiling computing device configured to compile said configuration states and paths;

(b) a tool or work station provided to compile said configuration states and paths;

(c) a computing running the task compiled from said configuration states and paths;

(d) a debugging hardware;

(e) a communication link;

(f) a printer configured to print a program represented by said configuration states and paths; and

(g) a display device configured to display elements of said configuration states and paths.

155. (Original) The programming method of claim 154 wherein said computing device is defined as a first computing device; said programming method further comprising a step to receive digital data representing the aforementioned steps through a communication link from a second computing device locating remote from said first computing device.

156. (Original) The programming method of claim 154 further comprising a step to specify one or more configuration states to comprise an output configuration.

157. (Original) The programming method of claim 156 wherein said output configuration defines the output conditions of one or more output terminals of said computing device.

158. (Original) The programming method of claim 156 wherein said output configuration defines a virtual computing output generated by said computing device.

159. (Original) The programming method of claim 154 further comprising a step to transform the specifications of the configuration states and the paths into digital data executable by said computing device.

160. (Original) The programming method of claim 154 further comprising a step to transform at least part of the specifications of the configuration states and paths with a secondary programming language of different format.

161. (Original) The programming method of claim 154 wherein the x configuration states and y paths are not necessary to be listed in sequential relationship to each other.

162. (Original) The programming method of claim 154 further comprising a step to provide a keyword for identifying the location of the program composed by said steps.

163. (Original) The programming method of claim 154 further comprising a step to group the configuration states of step (2) into one or more tables, and one of the configuration states in each table is specified to be active.

164. (Original) The programming method of claim 154 further comprising a step to organize a separated table to determine if the program composed by steps (2) to (6) is active or not active.

165. (Currently amended) The programming method of claim 154 further comprising the following steps:

(~~8~~ 9) provide a predefined instruction set to program the paths and configuration states;

(~~9~~ 10) define alternate expression to represent a specific instruction of the instruction set of step (8);

(~~10~~ 11) compose program with the alternate expression and

(~~11~~ 12) configure said program to be executable by said computing apparatus.

166. (Currently amended) The method of claim 165 wherein the process of step (~~11~~ 12) is a transformation process carried out by an editor, a compiler, an interpreter or a translation program.

167. (Original) The method of claim 165 further comprising a step to provide a table for cross reference between said alternate expression and the corresponding specific instruction.

168. (Original) The method of claim 154 further comprising a step to specify a configuration state to be active.

169. (Original) The method of claim 154 further wherein a path is defined as a default path to be executed at initialization.

170. (Currently amended) A programming method to facilitate a programmer to identify predetermined custom expressions in the program listing of a ~~programming tool~~ program for programming a computing device, said programming method comprises the steps of:

(1) defining one or more sets of custom expressions;

(2) equating the custom expressions of step 1 with a reference set of expressions;

(3) selecting a desirable set of expressions from a predetermined number of different expression sets derived from the steps (1) and (2);

(4) composing said program listing with the selected expression set; and

(5) transforming said program listing into codes executable by said computing device ;and

(6) providing at least one of the following hardware to work with said steps (1) to (5):

(a) a compiling computing device configured to compile said program;

(b) a tool or work station provided to compile said program;

(c) a computing running the task compiled from said program;

(d) a debugging hardware;

(e) a communication link;

(f) a printer configured to print at least a part of said program; and

(g) a display device configured to display elements of program.

171. (Currently amended) The programming method of claim 170 further comprising the steps of

(~~6~~ 7) defining x configuration states, at least one of said configuration states defines one or more qualifying conditions;

(~~7~~ 8) define y paths to be executed by said computing device; and

(~~8~~ 9) assign one of the paths to be executed by said computing device when a specific qualifying condition is satisfied.

172. (Original) The programming method of claim 170 wherein said custom expression set is defined by a table having a keyword.

173. (Original) The programming method of claim 170 wherein said predetermined expression set is configured to highlight user defined labels.

174. (Original) The programming method of claim 170 further comprising a step to select one of the different ways to highlight said selected custom expression set.

175. (Currently amended) A programming method for a program tool to program a computing device responsive to one or more qualifying conditions, to execute one or more paths; said programming method minimally comprises the steps of:

(1) defining a programming objective;

(2) specifying x configuration states in accordance to said programming objective wherein x is an integers equal or greater than one;

(3) specifying one or more qualifying label to at least one of said x configuration states,

(4) directing the qualifying label of step 2 to a separated expression describing the qualifying condition or conditions represented by said label;

(5) specifying y executable paths in which at least one path is executed in response to a qualifying condition specified by step (4);

(6) transforming said configuration states and paths into codes executable by said computing device; and

(7) providing at least one of the following hardware to work with said steps (1) to (6):

(a) a compiling computing device configured to compile said configuration states and paths;

(b) a tool or work station provided to compile said configuration states and paths;

(c) a computing running the task compiled from said configuration states and paths;

(d) a debugging hardware;

(e) a communication link;

(f) a printer configured to print a program represented by said configuration states and paths; and

(g) a display device configured to display elements of said configuration states and paths.

176. (Original) A first computing apparatus connected with a second remote computing device comprising:

computing means to execute a program; and

memory means storing digital data executable by said first computing apparatus or said second remote computing device; wherein said digital data comprising representation of a table format program having x configuration states and y paths; at least one of said configuration states defines one or more qualifying conditions; and one of said path is executed by said computing means when a specific qualifying conditions is satisfied.